ULTRASONIC TIMER SYSTEM

BY

MUSTAFA AL HOMSI

FACULTY ADVISOR: KENNETH J. BREEDING

Department of Electrical and Computer Engineering

Submitted in partial fulfillment of the requirements for the senior project design at the
Electrical and Computer Engineering department
in the college of Engineering of the
Ohio State University, 2005

Columbus, Ohio

Table of Contents

1. Abstract

This paper is intended to explain the ultrasonic timer project for the senior design project class ECE H683 winter quarter 2005. The paper includes the description and schematics for each component used, also the final source code is attached at the end.

2. Introduction

In the past years, the OSU football team had asked the Electrical Engineering department if it is possible to design a device that times how fast the football players are while training. The idea was simple. The player will be timed as soon as he starts running, and the timer would stop automatically when the runner reaches the finish line. A group of electrical engineering students had already designed the device. The design of the timer unit was straightforward. However, the implementation of the finish line detection was challenging. The group chooses to use infrared light for their finish line detection mechanism. This decision led to two main problems. The first is the necessary alignment of the infrared sensors. The second problem is the great influence of the surrounding weather conditions on infrared waves. As a result, the OSU football team was unable to benefit from the design as they were expecting. By replacing infrared sensors with ultrasonic ones, I am proposing a possible solution for the previous problems. Ultrasonic waves are capable of detecting objects without the need of providing a direct line of sight between the sender and the receiver. In addition, it is much less affected by the surrounding environment.

3. Objective

The main objective of my senior project is to design and develop a fully automatic athletic timer using ultrasonic sensors. The timer is very similar to a typical stopwatch timer with the added feature of automatic finish line detection. After starting the timer, the unit will keep advancing the time until the first runner reaches the finish line. When this happens, the timer will stop and output the final result. The design mainly consists of tow units:

- *Timer Unit:* Responsible for timing.
- *Finish Line Detection Unit:* Responsible for detecting passing objects through the finish line.

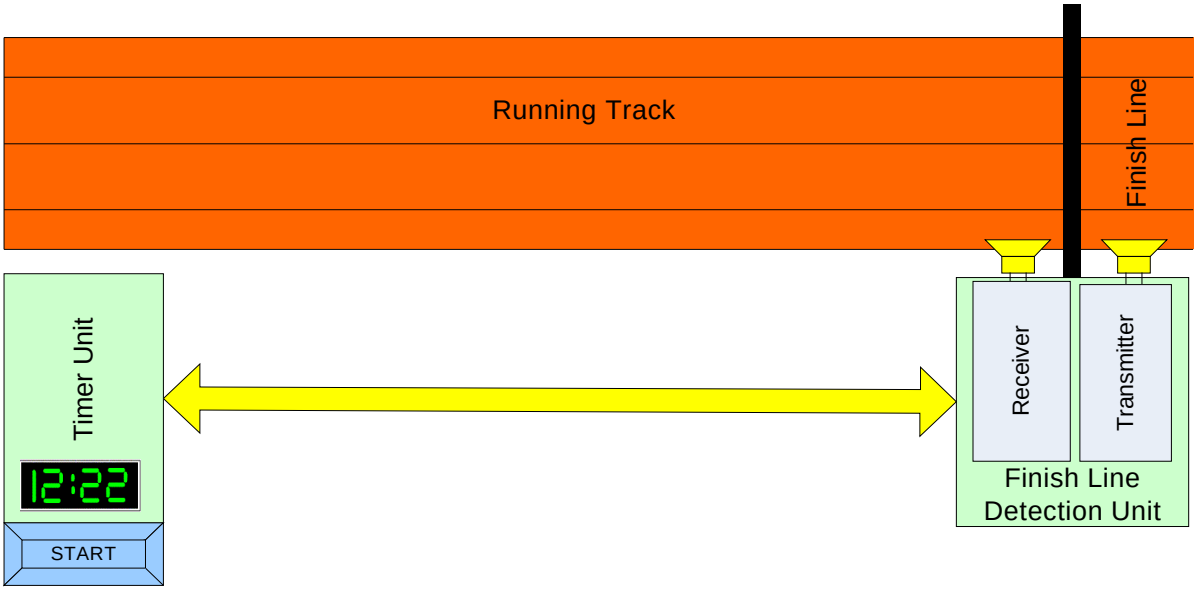

Figure 1: Main System Top View

4. Timer Unit

This unit starts the main timer when someone hits the start button and keeps time until a runner reaches the finish line, which would be detected by the finish line detection unit. Therefore, the unit is responsible for the following main tasks:

- Checking for the start button.
- Starting the Timer (When start button is pressed)
- Stopping the timer (When a runner reaches the finish line)
- Displaying the final result

The easiest way to implement the timer unit is to use a microcontroller. A simple assembly code is written for this purpose, and the following flow chart illustrates the flow of the program:

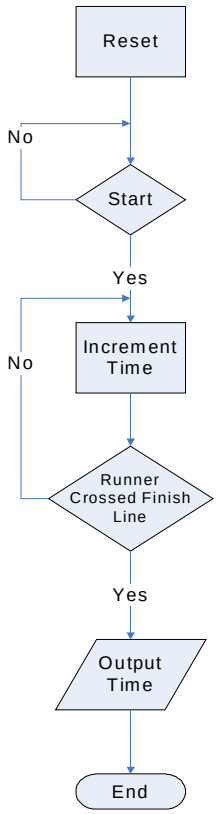

Figure 2: Main Flow Chart

5. Finish Line Detection Unit

This unit detects whether a runner has reached the finish line. When someone crosses the finish line, the output of the unit should change to trigger the timer unit. Otherwise the output should stay stable. In the design, ultrasonic signal was selected to provide the finish line detection. The idea is very simple. Two ultrasonic sensors (transmitter and receiver) would have to be placed at the finish line as shown in figure 3:

Figure 3: Running Track with the Sensors

The transmitter will keep on sending ultrasonic pulses. If there was not any object in the direct path of the ultrasonic transmitter, there will not be any reflected signals and the receiver will detects nothing, indicating to the timer unit that nobody has crossed the finish line yet. However, as soon as a runner crosses the ultrasonic pulses sent by the transmitter, some of those pulses will be reflected out of his or her body, and the receiver will detects those signals indicating to the timer unit that someone has crossed the finish line.

Before we can go deep into the details implementations of the finish line detection unit, first the basic fundamentals of ultrasonic sensors will be introduced.

5.1. Ultrasonic Sensors

For this design, two air transmission ultrasonic sensors using piezo ceramic elements were used to transmit and receive ultrasonic sound in the air. Since ultrasonic sensors responsible for transmitting and receiving signals, there are two types of these sensors: ultrasonic transmitter and ultrasonic receiver.

5.1.1 Ultrasonic Transmitter

The piezo ceramic element of our sensor has a resonant frequency of 40 kHz. When the ultrasonic transmitter is driven by signal that has the same frequency of the piezo ceramic element resonant frequency, the piezo ceramic element will vibrate transmitting ultrasonic waves of the same frequency.

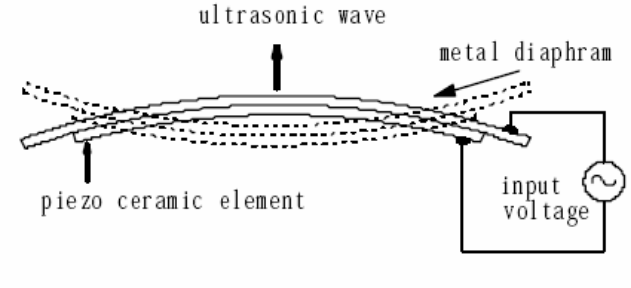

Figure 4: Ultrasonic Transmitter

5.1.2 Ultrasonic Receiver

The ultrasonic receiver has also a piezo ceramic element, but in this sensor the element will vibrate only if there are ultrasonic waves in the surrounding air that has similar frequency to the resonant frequency of the ceramic element. The output of the

sensor will be an alternating signal that has the same frequency of the received ultrasonic signal.

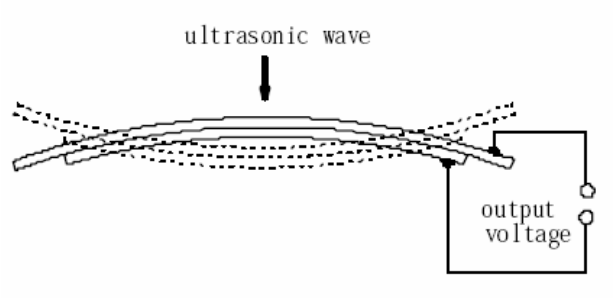

Figure 5: Ultrasonic Receiver

After understanding the fundamentals of the ultrasonic sensors, it becomes clear that in order to detect an object at the finish line using ultrasonic signals, we need to have at least two ultrasonic sensors driven by two circuits:

- *Transmitter Circuit*: Responsible for driving the ultrasonic transmitter.
- *Receiver Circuit:* Responsible for driving the ultrasonic receiver.

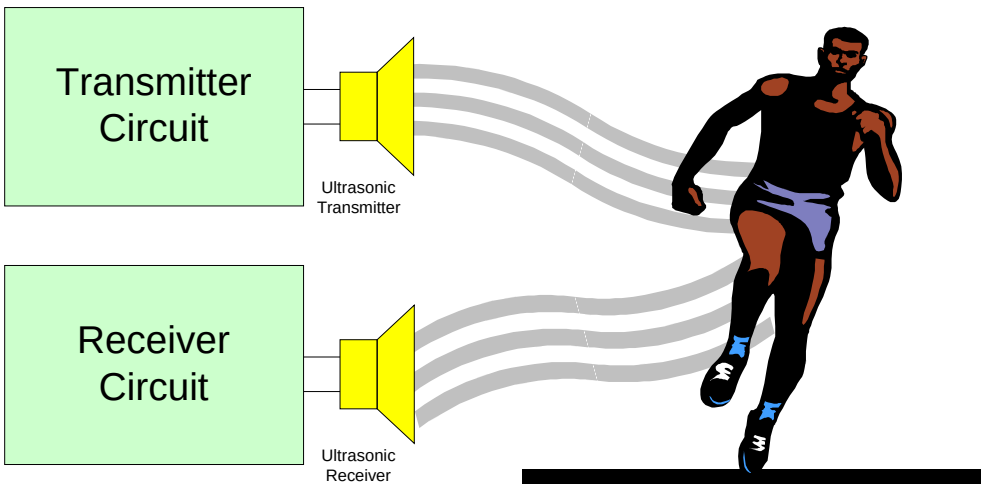

Figure 6: Transmitter and Receiver Circuit

5.2. Transmitter Circuit

To drive the ultrasonic transmitter sensor, we need to have a 40 kHz square signal range between –5 V and +5 V. The easiest way to do this is to use the pulse modulator function of the microcontroller to produce a 40 kHz square signal range from 0 V (logical 0) to 5 V (logical 1). In order to vary the signal from –5 V to +5 V, two NPN transistors were added to produce the exact same square signal but with different phase of 180°. The inverters are used to invert the voltage level. The two signals are then applied to the sensor.

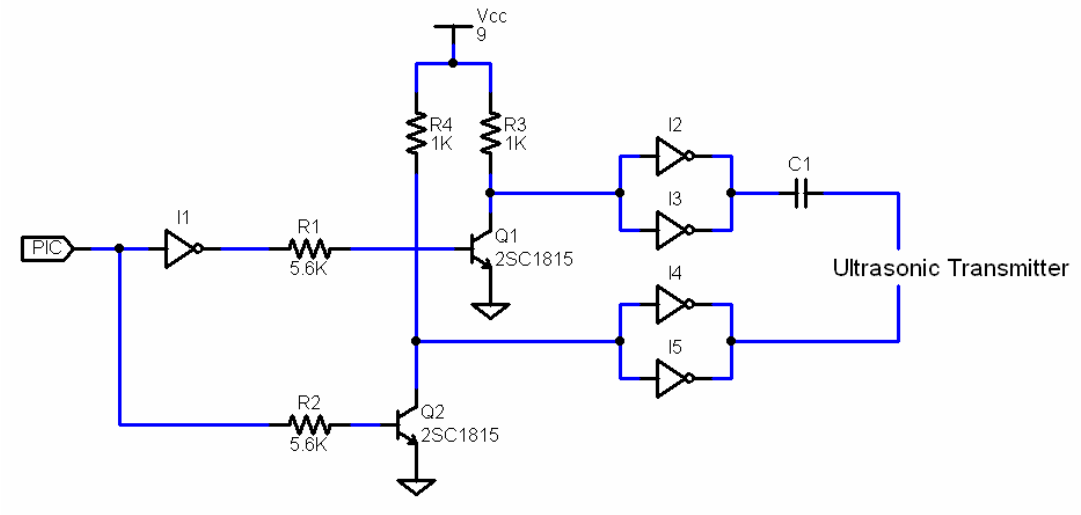

Figure 7: Ultrasonic Transmitter

5.3. Receiver Circuit

After the ultrasonic detects an ultrasonic signal, the sensor will output a 40 kHz signal but with a very small amplitude. Therefore, an input amplifier stage is required to amplify the small amplitude output of the ultrasonic receiver. At the output of the amplifier circuit, we will have an amplified AC signal with 40 kHz frequency. In order to get rid of the AC sweep, the signal is passed through a rectifier to output a DC voltage

that is directly proportional to the amplitude of the amplified signal. At the end, a comparator circuit is necessary to convert all DC voltages higher than a reference voltage to logical 1 and ignore all small DC voltages produced by noise and convert them to logical 0. The following diagram illustrates the component of the receiver circuit:

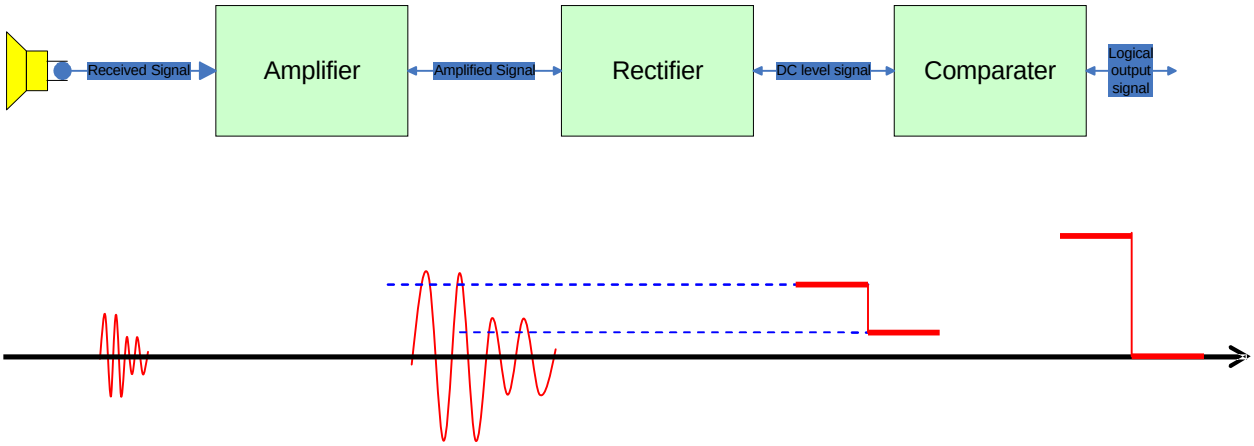

Figure8: Receiver Circuit

5.3.1 Amplifier Circuit

The amplifier circuit consists of two amplifying stages. The first stage implemented by using a standard op-amp, and a couple of resistors and capacitors. According to the resistor values shown in figure 9, this stage has a gain of 100. The second stage is implanted exactly in the same way, except that it has a different forward resistor value that decreases the gain to 10. The overall gain of the input amplifier circuit is 1000.

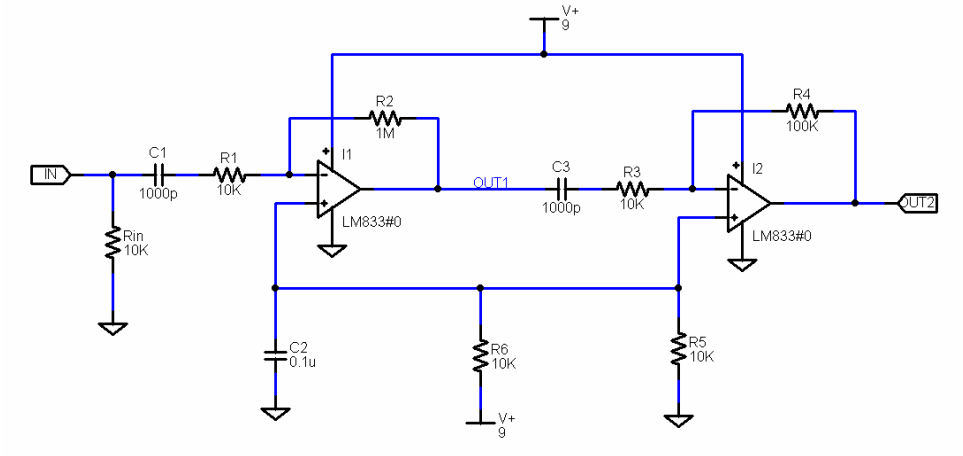

Figure 9: Amplifier Circuit

5.3.2 Rectifier Circuit

This is a simple rectifier circuit that consists of two Shottchy barrier diodes and a single capacitor. The first diode is connected in reverse to act as a Zener diode protecting the rest of the circuit from high voltages. The second diode and the capacitor rectify the AC sweep of the input signal and produce a DC level that directly related to the AC amplitude of the input signal.

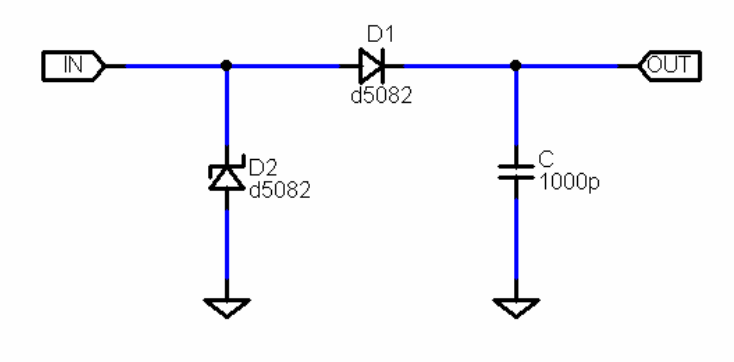

Figure 10: Rectifier Circuit

5.3.3 Comparator Circuit

The DC output of the rectifier circuit has to be compared with a reference voltage level to distinguish between detected and not detected situations and to ignore any noise input. For this reason, a low noise op-amp is implemented as shown in figure 10. The reference voltage is produce by the voltage divider and connected to the negative input of the op-amp. The input signal is directly connected to the positive input of the op-amp. Both voltages compared. If there is difference in voltage between the two inputs, this difference is amplified by the extreme high gain of the op-amp and outputted on the output node of the op-amp. Therefore, if the positive input (input signal) higher than the negative input (reference signal), a high voltage, close to the positive supply voltage, is produced on the output node; otherwise, we will have a voltage close to the negative voltage supply and in our case a 0 V is produced on the output node. Since we are using +9V for the positive supply voltage, we have to add a voltage divider at the end of the comparator circuit to convert it to TTL logic (5V corresponds to logical 1).

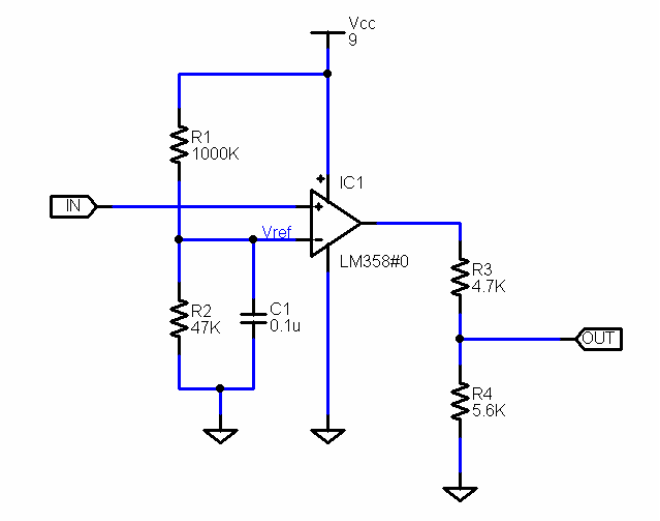

Figure 11: Comparator Circuit

The final diagram of the finish line detection unit is as the following:

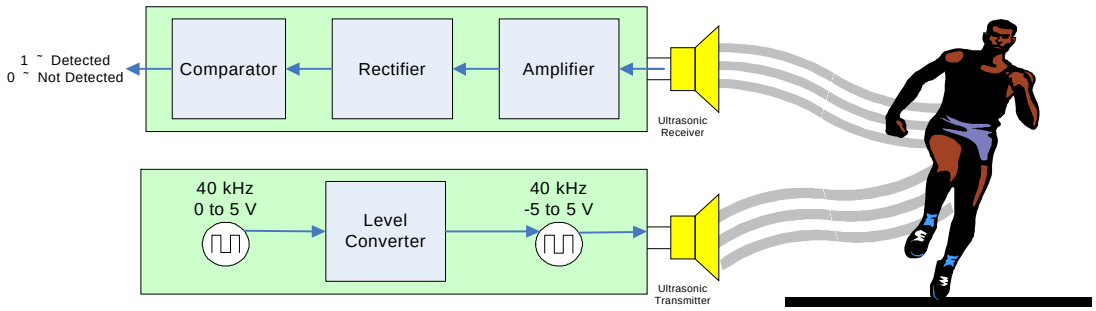

Figure 12: Finish Line Detection Circuit

Even though the above unit drives the ultrasonic transmitter and receiver perfectly, the unit will not function as expected. The reason for that is the influence of the ultrasonic transmitter on the receiver. In other word, while the transmitter is sending ultrasonic pulses, the receiver detects those pulses before it even reflect on any object. In order to solve this problem, we have to send the ultrasonic waves for specific amount of time while disabling the receiver circuit. After this specific time passed, we stop sending any ultrasonic waves and enable the receiver circuit for detection. The following figure and the flow chart illustrate the idea:

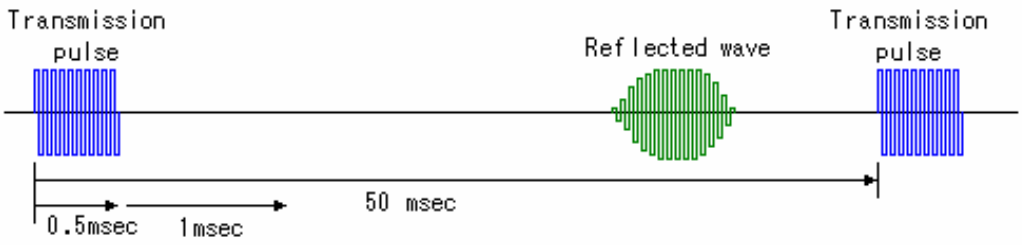

Figure 13: Receiver and Transmitter Waveforms

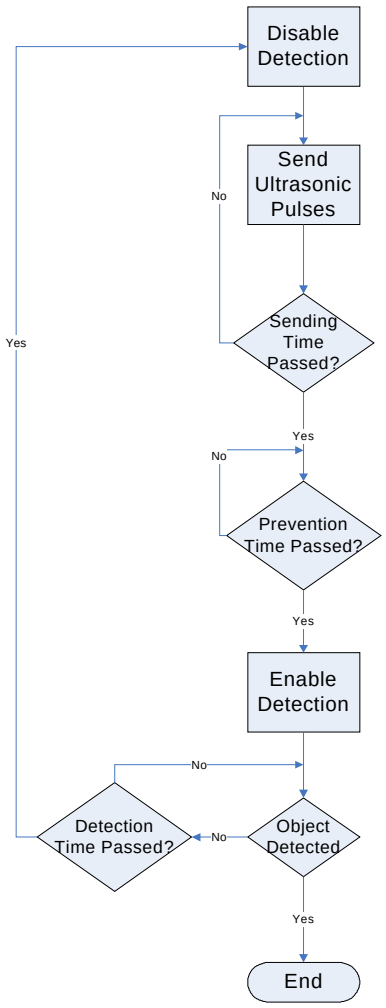

Figure 14: Detection Flow Chart

The ultrasonic signal sending period is set to 0.5 msec which is enough to send ultrasonic signals to the maximum range of the sensor. 1 msec for the prevention time period is reasonable to prevent any influence of the transmitter. Also, 48.5 msec is enough to detect reflected signal coming from any object located within the maximum range of the sensor.

In order to implement the previous flow chart for detecting an object, a mechanism for enabling and disabling the detection should be provided. The next section illustrates how this is accomplished.

5.3.4 Enable/Disable Detection Circuit

An S-R latch was used to implement the enabling and disabling of the detection. A control signal coming from the PIC processor which used to control the enabling and disabling of the detection is combined with the receiver circuit output and connected to the S-R latch input as shown in the following figure:

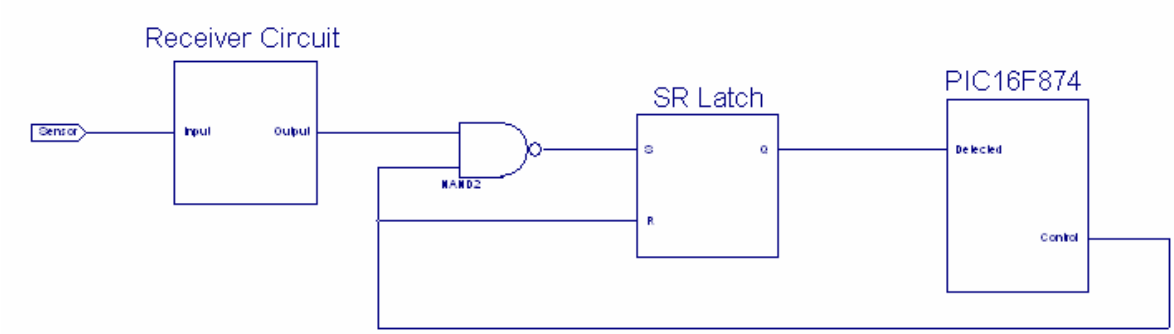

Figure 15: Enable/Disable Detection Circuit

After mapping the signals as shown above, the circuit can enable/disable the detection by changing the value of the control signal. The following truth table illustrates the idea:

Control	Input	S = Control NAND Input	R = Control	Q'	Detected
0	0	1	0	0	0
0	1	1	0	0	0
1	0	1	1	0	0
1	1	0	1	1	1

Table 1: Enable/Disable Circuit Truth Table

At the end, the final diagram for the finish line detection unit and the timer unit looks like:

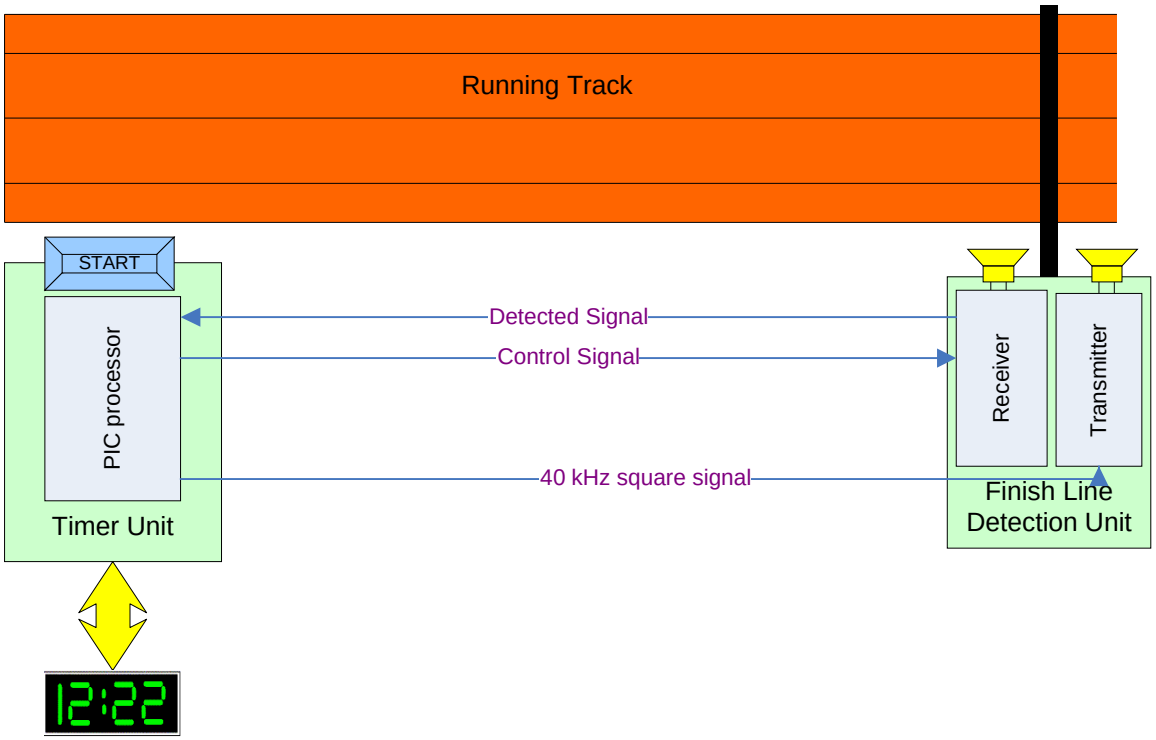

Figure 16: Final Diagram

6. Source Code Explanation

The final source code is included in Appendix A. In this section, the code and its components will be described. The figure below shows the main flow chart of the code:

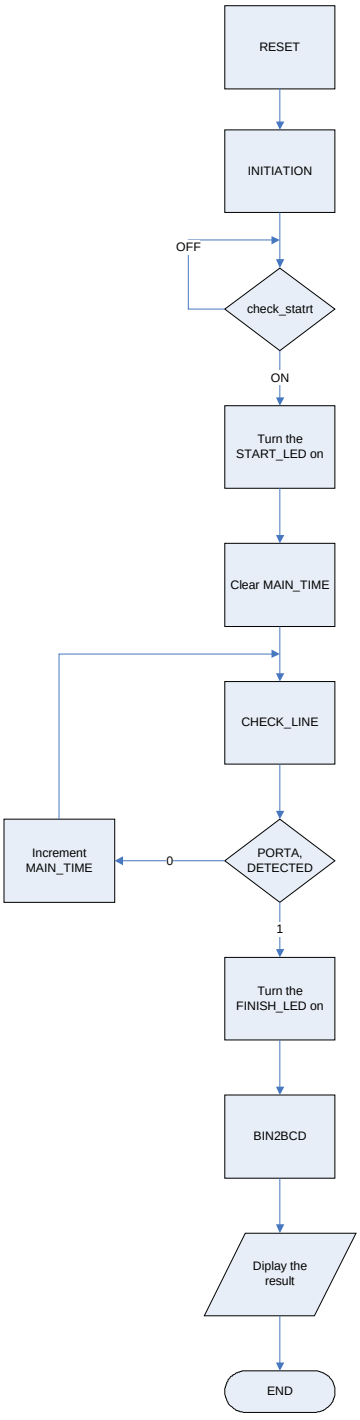

Figure 17: Source Code Flow Chart

After the power is on, the program calls the subroutine INITIATION to initiate the PIC processor. Then, the code checks if someone has pressed the start button. As soon as this is done, the controller turns on the START_LED, clears the MAIN_TIME registers, and calls the subroutine CHECK_LINE to check if someone has crossed the finish line. If no one has yet crossed the line, the controller advances the MAIN_TIME and calls CHECK_LINE again. However, if someone did cross the line, the controller turns on the FINISH_LED, calls the BIN2BCD routine, and displays the final result on a separate LCD. The LCD had an on-chip controller, so it was very easy to interface. It is clear from the main flow chart and the above description that the code calls three important subroutines: INITIATION, CHECK_LINE, and BIN2BCD.

6.1 INITIATION

This routine clears TIMR0, and calls another four subroutines to initiates PORTC, PORTB, PORTD and TIMR0.

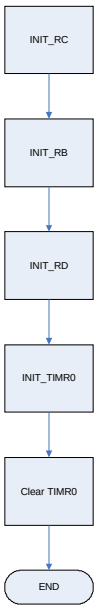

Figure 18: INITIATION Flow Chart

6.1.1 INIT_RC

PORT C is initialized so that RC<1:0> are input bits, and RC<5:2> are outputs and they are connected as follow:

RC0 ß START_BUTTON

RC1 ß DETECTED

RC2 à CONTROL

RC3 à START_LED

RC4 à FINISH_LED

RC5 à ULTRA_SEND

6.1.2 INIT_RB

Only the first three bits of PORT B are used. Those bits (RB<2:0>) are configured as input bits, and they are connected to the RS, RW, ENA pins of the LCD controller.

6.1.3 INIT_RD

This routine initializes PORT D as an output port and is connected to the input port of the LCD controller.

6.1.4 INIT_TIMR0

The routine sets the prescaler value to 256, and disables all interrupts.

6.2 CHECK_LINE

This routine is responsible for controlling the finish line detection unit. The following flow chart illustrates the flow of control:

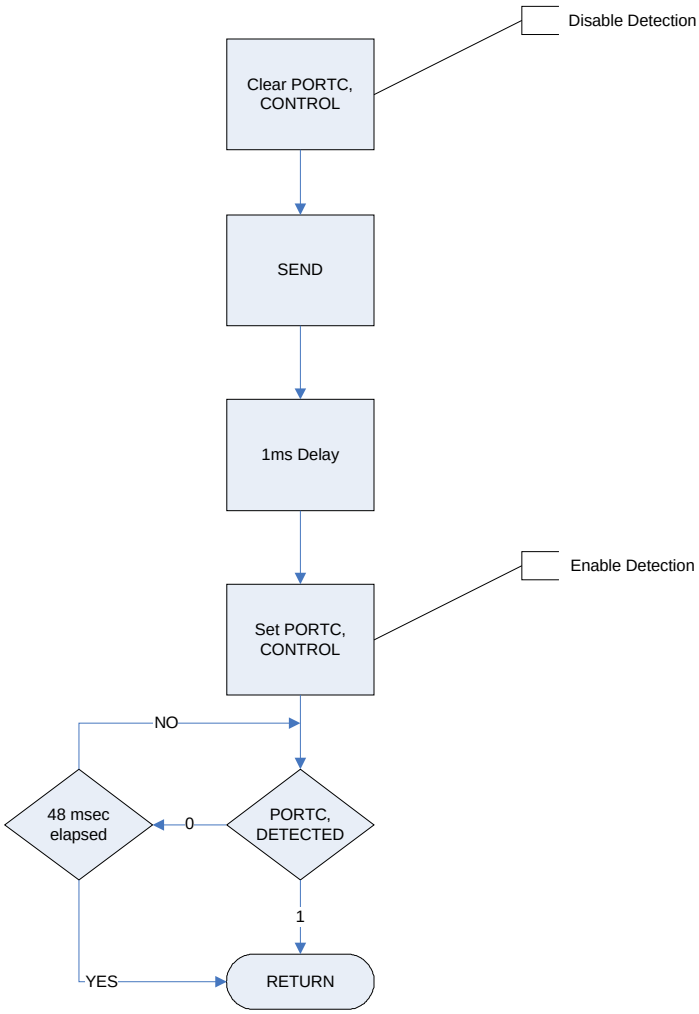

Figure 19: CHECK_LINE Flow Chart

The code first disables the detection by clearing the CONTROL bit in PORT C. Then, the SEND routine is called which will send 20 square wave pulses on bit ULTRA_SEND for 0.5 msec to drive the ultrasonic transmitter. After sending the 20 pulses, the code waits for 1 msec to prevent any influence from the transmitter. After the

1 msec elapsed, the detection is enabled again and the code starts checking the DETECTED signal. If there is an object detected, the code will exist. Otherwise the code checks if 48 msec has elapsed. If so, the code exist the routine. If not, the code keeps checking for the DETECTED signal again.

6.2.1 SEND

The SEND routine benefits from the pulse modulation feature of the PIC controller. The PIC controller has to be configured so that it produces a square pulse every 25 μs. Therefore, the CCPR1L and PR2 registers have to be initialized to 12 and 24 consequently so we have a 25 μs pulse with 50% duty cycle. The total execution time of the SEND routine is 25 μs × 20 pulses = 0.5 msec.

6.3 BIN2BCD

This routine converts the binary value to BCD values and stores the result in BCD1 and BCD2 register.

7. Conclusion

I plan on successfully completing the project within the end of winter quarter 2005 and have a final working design. We are currently on schedule for the completion of this project. Our development will not only overcome the difficulties of the finish line detection but it will supply an accurate and diverse means of providing times for training purposes. The mobility provided by the unnecessary alignment of the ultrasonic receiver and transmitter is unparalleled.

APPENDIX A

```
;*****************************************************************************
;                                                                             *
;          Project: ULTRASONIC TIMER SYSTEM                                   *
;    Filename: UltrasonicTimer.asm                                            *
;    Author: Mustafa Al Homsi                                                 *
;    Faculty Advisor: Prof. Kenneth Breeding                                  *
;                                                                             *
;*****************************************************************************

        list    p=16f874        ; list directive to define processor
        #include <p16f874.inc>      ; processor specific variable definitions

        __CONFIG _CP_OFF & _WDT_ON & _BODEN_ON & _PWRTE_ON &
_RC_OSC & _WRT_ENABLE_ON & _LVP_ON & _CPD_OFF

;*****************************************************
;***************VARIABLE DEFINITIONS******************
;*****************************************************
;DELAIES
D1ms            EQU         -D'4'
D5m             EQU         -D'20'
D15m            EQU         -D'59'
D48ms           EQU         -D'187'

;RC_INPUTS
START_BUTTON    EQU         0
DETECTED        EQU         1

;RC_OUTPUTES
ULTRA_SEND      EQU         3
CONTROL         EQU         4
START_LED       EQU         5
FINISH_LED      EQU         6

;PWM
T               EQU         D'24'
DUTY_CYCLE      EQU         D'12'
PWM_PIN         EQU         2
PULSE_COUNT     EQU         0x40

;MAIN_TIMER
LOOP_COUNT      EQU         0x20
MAIN_TIME1      EQU         0x21
MAIN_TIME2      EQU         0x22

;BIN2BCD
BCD1            EQU         0x22
BCD2            EQU         0x23

;LCD
RS              EQU         0
RW              EQU         1
ENA             EQU         2
```

```
LCD_COUNT       EQU     0x30
OFF             EQU     0X08
ON              EQU     0X0C
CLR             EQU     0X01
FSET            EQU     0x38
MODSET          EQU     0X06
ADDR            EQU     0x31
DIGIT           EQU     0x32
DIGIT1          EQU     0x33
DIGIT2          EQU     0x34
DIGIT3          EQU     0x35
DIGIT4          EQU     0x36
;*****************************************************

                ORG     0x0000
RESET GOTO      MAIN

                ORG     0X0004
INTHND          RETFIE

                ORG     0x0020
                GOTO    MAIN
;*****************************************************

INIT_RC
                CLRF    PORTC

                BSF     STATUS, RP0         ;BANK 1

                MOVLW   B'00000011'
                MOVWF   TRISC               ;RC0 = START_BUTTON
                                            ;RC1 = DETECTED
                                            ;RC3 = ULTRA_SEND
                                            ;RC4 = CONTROL
                                            ;RC5 = START_LED
                                            ;RC6 = FINISH_LED

                BCF     STATUS, RP0  ;BANK0
                RETURN
;*****************************************************

INIT_RB
                CLRF    PORTB

                BSF     STATUS,RP0          ;BANK1

                MOVLW   B'11111000'
                MOVWF   TRISB

                MOVLW   B'11111111'
                ANDWF   OPTION_REG,1

                BCF     STATUS,RP0
                RETURN
;*****************************************************
```

```
INIT_RD
                CLRF    PORTD

                BSF     STATUS,RP0      ;BANK1
                CLRF    TRISD

                BCF     STATUS,RP0
                RETURN
;*****************************************************

INIT_TMR0
                BSF     STATUS, RP0     ;BANK 1

                MOVLW   B'11000000'     ;TOCS=0, TOSE=0
                ANDWF   OPTION_REG, F
                MOVLW   B'00000111'     ;PSA2:PSA0 = 0 --> PSA = 256
                IORWF OPTION_REG, F

                BCF     STATUS, RP0     ;BANK 0
                CLRF    INTCON                  ;disable all interreupt,
TOIF=0
                RETURN
;*****************************************************

DELAY
                BCF     INTCON, T0IF
                MOVWF   TMR0
wait            BTFSS   INTCON, T0IF
                GOTO    wait
                RETURN
;*****************************************************

PWN
        ;Intilize PWN
                CLRF    CCP1CON
                CLRF    TMR2

        ;Set duty cycle
                MOVLW   DUTY_CYCLE
                MOVWF   CCPR1L

                CLRF    INTCON


                BSF     STATUS, RP0     ;BANK 1

        ;Set the period
                MOVLW   T
                MOVWF   PR2

        ;Set CCP1 pin as output
                BCF     TRISC, PWM_PIN  ;CCP1 = RC2

        ;Disable all peripheral interupts
                CLRF    PIE1
```

```
        ;Clear peripheral interrupt flag
                BCF             STATUS, RP0             ;BANK 0
                CLRF            PIR1

        ;Set PWM
                MOVLW           0x0C
                MOVWF           CCP1CON

        ;Start PWM
                BSF             T2CON, TMR2ON                   ;start TMR2, PSA = 1:1

next    BTFSS PIR1, TMR2IF
                GOTO  next

                BCF             PIR1, TMR2IF

                DECFSZ          PULSE_COUNT
                GOTO            next

                RETURN
;*****************************************************

SEND
                MOVLW           D'20'                   ;20 pulses
                MOVWF           PULSE_COUNT             ;20 x 1/40 KHZ = 1/2000 = 0.5
                CALL            PWN
                RETURN
;*****************************************************

CHECK_LINE
        ;Disable detection
                BCF             PORTC, CONTROL

        ;Send ultrasonic pulses for 0.5 msec
                CALL            SEND

        ;Prevent error detection for 1ms
                MOVLW           D1ms
                CALL            DELAY

        ;INIT_64ms
                BCF             INTCON, T0IF
                MOVLW           D48ms
                MOVWF           TMR0

        ;Enable detection
                BSF             PORTC, CONTROL

        ;check if someone crossed the finish line
check_again
                BTFSC           PORTC,DETECTED
                RETURN

        ;check if 64ms elapsed
                BTFSS           INTCON, T0IF
                GOTO            check_again
```

```
                RETURN
;*****************************************************

BIN2BCD
                CLRF        BCD1
                CLRF        BCD2

hundred         ADDLW       0x9C                ;-100
                BTFSS       STATUS,C
                GOTO        tenth

                INCF        BCD2
                GOTO        hundred

tenth     ADDLW             0x64

again     ADDLW             0xF6                ;-10
                BTFSS       STATUS,C
                 GOTO       swapBCD

                INCF        BCD1,F
                GOTO        again

swapBCD         ADDLW       0xA                 ;w + 10
                SWAPF       BCD1,F
                IORWF       BCD1,F
      RETURN
;*****************************************************

RST_LCD
                MOVLW       0X03
                MOVWF       LCD_COUNT

rst_loop
                BCF         PORTB,RS

                MOVLW       D15m
                CALL        DELAY

                BCF         PORTB,RW
                BSF         PORTB,ENA
                MOVLW       FSET
                MOVWF       PORTD

                NOP
                BCF         PORTB,ENA

                DECFSZ      LCD_COUNT,1
                GOTO        rst_loop

                RETURN
;*****************************************************

INT_LCD
                CLRF        LCD_COUNT
```

```
init_next
                MOVLW       D5m
                CALL        DELAY

                BSF         PORTB,ENA

                MOVF        LCD_COUNT,0
                CALL        DISP_COMMAND
                MOVWF       PORTD

                NOP
                BCF         PORTB,ENA

                INCF        LCD_COUNT,F     ;TEST IF COUNT=5
                MOVLW       0X05
                SUBWF       LCD_COUNT,0
                BTFSS       STATUS,Z
                GOTO        init_next

                MOVLW       D5m
                CALL        DELAY

                RETURN

;*****************************************************

LCD_DISP
                BCF         PORTB,RS        ;SET ADDR
                BCF         PORTB,RW
                BSF         PORTB,ENA
                MOVF        ADDR,W
                MOVWF       PORTD
                NOP
                BCF         PORTB,ENA


                MOVLW       D5m
                CALL        DELAY

                BSF         PORTB,RS        ;SET DIGIT
                BCF         PORTB,RW
                BSF         PORTB,ENA
                MOVF        DIGIT,W
                MOVWF       PORTD
                NOP
                BCF         PORTB,ENA

                MOVLW       D5m
                CALL        DELAY

                RETURN
;*****************************************************

DISP_COMMAND
                ADDWF       PCL,1
```

```
                RETLW       FSET
                RETLW       CLR
                RETLW       OFF
                RETLW       ON
                RETLW       CLR
;*****************************************************

DIGIT_TABLE
                ADDWF       PCL,F

                RETLW       A'0'
                RETLW       A'1'
                RETLW       A'2'
                RETLW       A'3'
                RETLW       A'4'
                RETLW       A'5'
                RETLW       A'6'
                RETLW       A'7'
                RETLW       A'8'
                RETLW       A'9'
                RETLW       A'.'
;*****************************************************

INITILIZATION
                CALL        INIT_RC
                CALL        INIT_RB
                CALL        INIT_RD
                CALL        INIT_TMR0
                CLRF        TMR0

                RETURN
;*****************************************************

;*****************************************************
;*******************MAIN PROGRAM**********
;*****************************************************
MAIN
                CALL        INITILIZATION

        ;Check for START button
check_start
                BTFSS       PORTC, START_BUTTON
                GOTO        check_start

        ;Turn on the START_LED
                BSF         PORTC, START_LED

        ;Clear the main timer
                CLRF        MAIN_TIME1
                CLRF        MAIN_TIME2

                MOVLW       0x02
                MOVWF       LOOP_COUNT

        ;Finish line detection
loop
```

```
                CALL    CHECK_LINE
                BTFSC   PORTC,DETECTED
                GOTO    finish
                DECFSZ  LOOP_COUNT
                GOTO    loop

                MOVLW   0x02
                MOVWF   LOOP_COUNT
                INCFSZ  MAIN_TIME1
                GOTO    loop

                INCF    MAIN_TIME2
                GOTO    loop

finish
        ;Turn on the FINISH_LED
                BSF     PORTC, FINISH_LED

        ;Convert the time to BCD
                MOVF    MAIN_TIME1,W
                CALL    BIN2BCD

                MOVF    BCD1,W
                MOVWF   DIGIT1
                MOVWF   DIGIT2

                MOVLW   0x0F
                ANDWF   DIGIT1,F
                SWAPF   DIGIT2
                ANDWF   DIGIT2,F

                MOVF    BCD2,W
                MOVWF   DIGIT3
                MOVWF   DIGIT4

                MOVLW   0x0F
                ANDWF   DIGIT3,F
                SWAPF   DIGIT4
                ANDWF   DIGIT4,F


        ;Display the result (huundreds and tens)
                BCF     PORTB,ENA
                BCF     PORTB,RW
                CALL    RST_LCD
                CALL    INT_LCD

        ;Hundreds
                MOVLW   0x80
                MOVWF   ADDR
                MOVF    DIGIT4,W
                CALL    DIGIT_TABLE
                MOVWF   DIGIT
                CALL    LCD_DISP

        ;Tens
```

```
        INCF    ADDR
        MOVF    DIGIT3,W
        CALL    DIGIT_TABLE
        MOVWF   DIGIT
        CALL    LCD_DISP

;ones
        INCF    ADDR
        MOVF    DIGIT2,W
        CALL    DIGIT_TABLE
        MOVWF   DIGIT
        CALL    LCD_DISP

;Diplay point
        INCF    ADDR
        MOVLW   0x0A            ;point
        CALL    DIGIT_TABLE
        MOVWF   DIGIT
        CALL    LCD_DISP

;Diplay thenths
        INCF    ADDR
        MOVF    DIGIT1,W
        CALL    DIGIT_TABLE
        MOVWF   DIGIT
        CALL    LCD_DISP

        GOTO    RESET
        END
```